COBB

INSIDE DOS®™

Tips & techniques for MS-DOS & PC-DOS Versions 5 & 6

November 1994
Vol. 5 No. 11

Examining the FORMAT command

VERSION 5.0 & 6.x

Formatting a disk—how could something so basic have become so complicated? A stroll through the history of the PC can answer this question. As the PC evolved, Microsoft and other vendors released new versions of DOS that were compatible with the latest hardware on the market. For example, when double-sided disk drives replaced single-sided drives, Microsoft released DOS 1.1 to accommodate them. Then, when IBM started producing the XT, Microsoft released DOS 2 to provide support for hard disk management. Next came IBM's AT with a 1.2 Mb drive. Microsoft responded with DOS 3, whose drivers could access the new drive.

In each case, as new drives and hard disks expanded the capabilities of computer systems, software vendors had to upgrade the operating systems as well. Just as important, DOS's FORMAT command, which prepares disks for use, had to maintain compatibility with old hardware while adding new options to keep up with new hardware. As a result, the FORMAT command has become so loaded down with options and switches that it's nearly incomprehensible.

Beside the changes that new hardware brought to disk formatting, DOS 5 introduced a new twist—when DOS formats a disk, it copies the root directory, the FAT, and even the original boot record to unused areas of the disk, and then reinitializes the directory of the FAT. As a result, you can successfully unformat a disk you've accidentally formatted and recover all the disk's information. (Prior to DOS 5, a format operation permanently wiped out all information stored on a disk.)

In this article, we'll discuss the command switches for the FORMAT command and take a close look at its operation. Throughout this article, bear in mind that the discussion of formatting and unformatting refers specifically to floppy disks—not to hard disks.

The FORMAT command

The syntax for DOS 5's FORMAT command is

```
format drive: [/V[:label]][/B|/S][/F:size][/Q][/U]
```

or

```
format drive: [/V[:label]][/B|/S][/T:tracks
      /N:sectors][/Q][/U]
```

or

```
format drive: [/V[:label]][/B|/S][/1][/4][/Q][/U]
```

or

```
format drive: [/B|/S][/1][/4][/8][/Q][/U]
```

In DOS 6.x, the FORMAT command's syntax is essentially the same, except for a [/c] switch that appears at the end of each of the above syntaxes. Let's briefly discuss the FORMAT command's options and switches by breaking them into their components.

The first parameter

```
drive:
```

is the letter of the drive you want to format. FORMAT will attempt to use the highest-capacity format the disk drive supports unless you specify otherwise.

The switch

```
/V[:label]
```

instructs DOS to write a volume label to the disk. By default, the FORMAT command will prompt you for a

IN THIS ISSUE

A Publication of The Cobb Group

label, so you don't need to use the /V switch unless you want to specify the label on the command line. If you specify the volume label on the command line, don't include any spaces.

In the days when most PCs had only floppy disks, software developers used the switch

```
/B
```

to prepare their distribution disks. At that time, most PC users installed the system files on their disks to make them bootable. The system files are position sensitive, and they must appear as the first two directory entries on the disk and as contiguous files in the first two files in the data area. The /B switch creates two dummy files that reserve the required disk area so you can install the operating system using the SYS command. In DOS 6.x, the /S switch eliminates the need for the /B switch; however, the /B switch is preserved for backward compatibility.

The switch

```
/S
```

copies the operating system to the disk you're formatting. DOS copies the two hidden system files—IO.SYS and MSDOS.SYS—as well as COMMAND.COM to the disk, making it bootable.

Rarely used today, the switch

```
/1
```

tells DOS to format only one side of a floppy disk, even when you're using a two-sided drive. This switch allows you to format a 5.25-inch disk so that it's compatible with the old 160 Kb and 180 Kb formats. Both single- and double-sided drives can read single-sided disks.

The switch

```
/4
```

formats a 360 Kb disk in a 1.2 Mb 5.25-inch drive. DOS still fully supports this switch, which provides the quickest method of specifying this type of format. Incidentally, this switch is not equivalent to the /f:360 switch or to the /n:9 and /t:40 switches.

You use the switch

```
/8
```

to tell DOS to use only the first eight sectors of data on each track of a 5.25-inch disk. DOS 1's disk format program created only eight sectors per track. The /8 switch creates disks that the older operating system can read. Note that DOS will still write nine sectors

INSIDE DOS™

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices:

Domestic	$49/yr	($5.95 each)
Outside US	$69/yr	($8.50 each)

Phone:

Toll free	(800) 223-8720
Local	(502) 493-3300
Customer Relations Fax	(502) 491-8050
Editorial Department Fax	(502) 491-3433

Address: You may address tips, special requests, and other correspondence to

The Editor, *Inside DOS*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Postmaster: Second class postage is pending in Louisville, KY. Send address changes to

Inside DOS
P.O. Box 35160
Louisville, KY 40232

Authorized Canada Post International Publications Mail (Canadian Distribution) Sales Agreement #XXXXXX CANADA GST #123669673. Send returns to Canadian Direct Mailing Sys. Ltd., 920 Mercer Street, Windsor, Ontario, N9A 7C2. Printed in the USA.

Staff:

Editor-in-Chief	Jim Stanfield
Contributing Editor	Van Wolverton
Editors	Linda Watkins
	Tiffany Tayor
	Tim Hampton
Production Artists	Tara Dickerson
	Tre Pryor
Managing Editor	Bob Artner
Circulation Manager	Tammy Castleman
Editorial Director	Jeff Yocom
Publishers	Mark Crane
	Jon Pyles
President/CEO	Tom Cottingham

Advertising: For information about advertising in Cobb Group journals, contact Tracee Bell Troutt at (800) 223-8720, ext. 430.

Back Issues: To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $5.95 each, $8.50 outside the US. You can pay with MasterCard, VISA, Discover, or American Express, or we can bill you.

Bulk Sales: For information about bulk/group subscription sales, contact Larry Bolton at (800) 223-8720, ext. 387.

per track during a physical format operation, but it will ignore one of those sectors during a logical format operation.

The two switches

```
/T:tracks
/N:sectors
```

give FORMAT the ability to deal with disk formats introduced after the release of the operating system. There's only one major drawback to using these switches: For each disk format, you must supply the number of tracks and sectors per track—tedious information that's nearly impossible to remember.

The /F switch, whose form is

```
/F:size
```

accomplishes the same thing and is much easier to use. This switch simplifies the FORMAT command by allowing you to express the disk format in terms of storage capacity. Instead of having to remember how many sectors and tracks each disk format uses, you simply type the size of the disk. This is especially useful when you're formatting a disk to a capacity that's lower than the drive is capable of providing. For example, to format a 3.5-inch disk to 720 Kb in a 1.44 Mb drive, use the switch /F720. Table A lists the possible size parameters for each format.

New to DOS 5, the switch

```
/Q
```

disables the surface scan that the default safe format operation normally performs. The danger of using the /Q (quick) switch is that a sector might have become unreadable since the time you formatted the disk with the unconditional format (/U), which we'll explain next. If a sector becomes unreadable, DOS will not detect the problem when you format the disk with the /Q switch. Later, when you try to read data stored in a bad sector, DOS will generate a disk error, and you'll probably lose that data.

The switch

```
/U
```

makes the FORMAT command act as it did in versions prior to DOS 5, performing both a physical and logical format of the disk. This switch tells DOS to overwrite and verify all sectors of the disk and to mark any bad sectors as unusable. The /U switch might more appropriately be named the unrecoverable switch since it saves no data for the UNFORMAT command.

New to DOS 6, the switch

```
/C
```

tells DOS to retest any bad clusters instead of leaving them marked bad, as did previous versions of DOS.

The formatting operation

With a solid grasp of the format options behind us, let's examine how FORMAT operates. On an unformatted disk, the FORMAT command typically performs three major operations: It physically formats the disk; performs a simple surface scan to detect bad

Table A

Using this parameter...	specifies...
160, 160k, or 160kb	160 Kb, single-sided, double-density, 5.25-inch disk
180, 180k, or 180kb	180 Kb, single-sided, double-density, 5.25-inch disk
320, 320k, or 320kb	320 Kb, double-sided, double-density, 5.25-inch disk
360, 360k, or 360kb	360 Kb, double-sided, double-density, 5.25-inch disk
720, 720k, or 720kb	720 Kb, double-sided, double-density, 3.5-inch disk
1200, 1200k, 1200kb, 1.2, 1.2m, or 1.2mb	1.2 Mb, double-sided, quadruple-density, 5.25-inch disk
1440, 1440k, 1440kb, 1.44, 1.44m, or 1.44mb	1.44 Mb, double-sided, quadruple-density, 3.5-inch disk
2880, 2880k, 2880kb, 2.88, 2.88m, or 2.88mb	2.88 Mb, double-sided, 3.5-inch disk

You can specify the size of the floppy disk you wish to format by using one of the parameters in the first column.

Want to format a single disk quickly without seeing a lot of prompts? Then, use the little-known /AUTOTEST switch. This switch prevents the FORMAT command from displaying the *Insert new disk, volume label* and *Format another (Y/N)?* prompts.

sectors; and writes the boot record, root directory, and FAT to the disk.

If you formatted a disk previously but now want to format it with an option that will change its capacity, DOS will treat that disk as if it were unformatted. For example, if you format a disk to 360 Kb and then reformat it to 320 Kb, DOS will automatically perform a physical format.

The FORMAT command will act somewhat differently when you format a disk under the following conditions:

- you've already formatted the disk
- you haven't specified an unconditional format (/U)
- you aren't changing the disk's storage capacity

In these cases, FORMAT may perform all or some of these four operations:

- save the existing data
- initialize the directory and FAT
- perform a surface scan to identify bad sectors
- overwrite every data sector of the disk

Let's briefly discuss the format operation as it applies to previously formatted disks.

Physical and logical formatting

When you format a disk, the FORMAT program actually performs two formatting operations: a physical format and a logical format. When DOS performs a physical format, it begins by dividing the disk into two surfaces, or sides. (Nearly all disk drives you'll encounter today are double-sided and use both sides of the disk for storage.) DOS then divides each side into concentric rings called *tracks*. Finally, DOS divides the tracks into sections called *sectors*.

When the physical format is complete, the disk is still empty—it simply has the framework onto which you can store information. With this new framework in place, the computer's BIOS can read from and write to the disk.

Now that the formatted disk is BIOS-compatible, the next step is making it DOS-compatible. This process is called a logical format. This type of format consists of writing onto the disk several key elements that dictate, at the DOS level, how information is to be stored. We discuss the logical format in the section "The Formatting Operation" in the accompanying article.

By default, DOS's FORMAT command performs a safe format that lets you recover your data using the UNFORMAT command. The first stage of a safe format creates a MIRROR file. This file contains all the information the UNFORMAT command needs in order to restore the disk to its original condition. After DOS creates the MIRROR file, it writes that file to unused sectors on the disk. Next, DOS erases all file entries in the root directory and clears both copies of the FAT. However, DOS doesn't alter any of the data sectors on the disk.

As the last step in a safe format, DOS performs a surface scan operation. During the scan, the FORMAT command calls a BIOS (Basic Input/Output System) routine that attempts to read the disk. FORMAT reads each sector with a separate call so that it can detect any error at the sector level. (You might hear the drive step from track to track as it does this.) This scan makes sure that the drive electronics can locate and read each sector on the disk. If FORMAT finds any unreadable sectors, it will mark the corresponding entry in the FAT with a special code to tell DOS not to store any data in those sectors.

The quick format (/Q) performs the same steps as the safe format but eliminates the surface scan. The quick format still creates the MIRROR file, saving copies of the root directory and FAT, and clears the original root directory and FAT. But, because the quick format doesn't request that the BIOS read each sector on the disk, the time it takes to complete the format is greatly reduced. The tradeoff, of course, is the confidence you can place in the disk's integrity.

Specifying the unconditional format (/U) causes FORMAT to treat the disk as if it were unformatted. When you perform an unconditional format on a disk that contains valid information, DOS won't save information that the UNFORMAT command needs and will completely overwrite every sector on the disk, making the original data completely unrecoverable. The unconditional format is automatically invoked (no matter what switches you specify) if the disk you're formatting has never been formatted or if you're changing its format.

When you combine the quick and unconditional formats, FORMAT performs the minimum amount of work necessary to convince DOS that the disk contains no files. FORMAT doesn't create a MIRROR file, save data, or perform a surface scan. In fact, the effect is almost the same as deleting all the files and subdirectories on the disk.

Conclusion

The complexity of the FORMAT command seems to increase with each version of DOS. In this article, we've tried to bring you up to date on the FORMAT command by explaining each of its current switches. We've also explained the operations FORMAT may perform. ❖

Deleting all the files in a directory

VERSION 5.0 & 6.x

You can delete all the files in a directory by using a command in the form

```
del directory\*.*
```

or

```
del directory.
```

where *directory* is the name of the directory whose files you want to delete. But did you know that you can delete all the files in a directory simply by entering only the name of the directory after the DEL command? For example, we deleted all the files from our COBB directory by entering the command

```
c:\>del cobb
```

If you want to remove the files from a subdirectory, you simply enter the path to the subdirectory's name after the DEL command. For example, to delete the files from the STEVE subdirectory of our COBB directory, we entered the command

```
c:\>del cobb\steve
```

and pressed Y in response to the DOS prompt

```
All files in directory will be deleted!
Are you sure (Y/N)?
```

Keep in mind that using this method will delete only the files from the directory. If you also want to remove the directory and you're using DOS 6.x, you should use the DELTREE command to remove the directory, its files, and all its subdirectories and their files. If you're using DOS 5, you can remove the directory by using the RD command after you've deleted all the files from the directory.

Use the most recent versions of DOS drivers

VERSION 5.0 & 6.x

If you use both Windows and DOS, you should use the most recent versions of HIMEM, EMM386, and SMARTDRV. Microsoft builds these files into both operating systems and updates them with each new release. For example, if you have DOS 5 and Windows 3.1, you should use the Windows versions of the three files because the Windows 3.1 files are dated March 10, 1992, and the DOS 5 files are dated April 9, 1991. However, the DOS 6.0 and 6.2 files are dated March 10, 1993, and September 30, 1993, respectively, so you should use those files instead of the ones from Windows 3.1.

For example, if you're using DOS 5 and Windows 3.1, you might currently load the EMM386 driver from your CONFIG.SYS file by entering the line

```
device=c:\dos\emm386.exe
```

However, you should specify the Windows 3.1 version of the driver by changing the line in your CONFIG.SYS file to read

```
device=c:\windows\emm386.exe
```

After you load the most recent version of a driver, you should notice an improvement in your system's performance. If you're not satisfied with the performance of the newer driver, you can always revert to the older driver by revising your CONFIG.SYS or AUTOEXEC.BAT file.

Partial recall

VERSION 5.0 & 6.x

No, we're not going to talk about a follow-up to Arnold's blockbuster movie. But, we are going to talk about something just as exciting (well almost)—using the [F2] key to save you time at the command prompt. Specifically, you can press [F2] and a letter that appeared in the last command you entered. When you do, DOS will redisplay the command up to (but not including) that letter. For example, suppose you enter the command

```
c:\> cd\batch\util
```

to switch to the directory named C:\BATCH\UTIL. Then, suppose you want to change to a directory named C:\BATCH\TEST. To do so, you simply press [F2] and type *u*. DOS will then redisplay the previous *change directory* command up to, but not including, the letter *u*, as shown below:

```
c:\> cd\batch\
```

Now, all you need to do is type *test* and press [Enter] to change directories. ❖

Transferring files between a laptop and a desktop PC

VERSION
6.x

Do you work on both a laptop and a desktop PC? If so, you may have thought you'd need a sophisticated communications program to connect your two systems in order to transfer files between them. Fortunately, this isn't the case.

If your laptop and desktop systems are running DOS 6.x, all you need is a parallel or serial cable to connect them. Then, you use a pair of DOS 6.x programs to transfer files between the two systems.

An overview of the procedure

DOS 6.x's INTERLNK.EXE and INTERSVR.EXE files let you connect your laptop and desktop via a cable so that you can use one system to access the disk drives or the printer on the other system. When you connect two systems in this way, one system is known as the *client* and the other system is known as the *server*. The system that shares its drives is the server, and the system that accesses the other system's drives is the client.

To copy files from the laptop's hard disk to the desktop's hard disk, you'll configure the laptop as the server and the desktop as the client. When you do, the laptop's hard disk will be available to the desktop. Once you've set up everything, you can copy files from your laptop to your desktop just as if your laptop's hard disk were a second hard disk in the desktop. Now, let's take a more detailed look at the procedure.

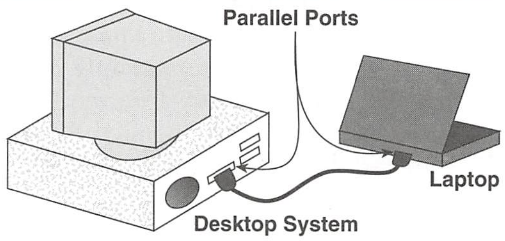
The cable choices

You can easily connect your two systems by using either a parallel or a serial cable. The type of cable you'll use depends on the ports available on your two systems. If you have serial ports free on both systems, you can use either a three-wire serial cable or a seven-wire null-modem serial cable. If you have parallel ports free on both systems, you can use a bidirectional parallel cable. You can obtain the necessary cables from your local computer store. (By the way, the parallel cable connection is faster than the three-wire connection, but the three-wire serial cable is thin enough to fit in your laptop's carrying case.)

The DOS 6.x programs

As we mentioned, INTERLNK and INTERSVR are the DOS 6.x programs you use to transfer files between your laptop and desktop. You use the INTERLNK program to set up the client system and you use the INTERSVR program to set up the server system. To see how these two programs work, let's suppose you have only one parallel port on your desktop (LPT1). You'll use a bidirectional parallel cable to connect LPT1 on the laptop to LPT1 on the desktop, as shown in Figure A.

Figure A

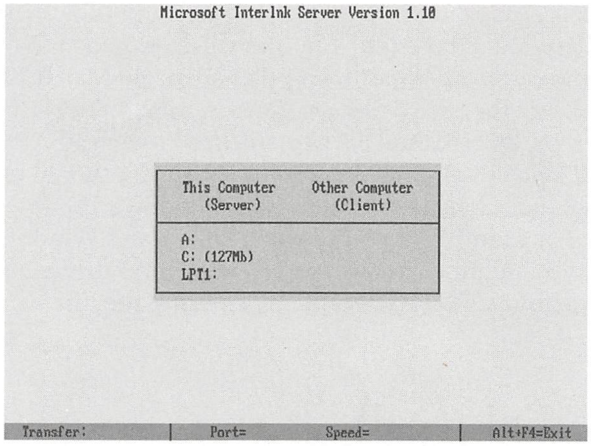


You connect LPT1 on your laptop to LPT1 on your desktop.

Setting up the laptop as the server

Setting up the laptop as the server is extremely easy. All you have to do is type *intersvr* at the DOS prompt. When you do, you'll see a screen similar to the one shown in Figure B.

Figure B



```
                 Microsoft Interlnk Server Version 1.10

              This Computer      Other Computer
                (Server)            (Client)

              A:
              C: (127Mb)
              LPT1:

 Transfer:        |    Port=        Speed=        |    Alt+F4=Exit
```

Running the INTERSVR command on the laptop sets up the server.

As you can see in the Server column, the laptop is ready to share both drives A and C. The Client column remains empty until you set up the desktop system.

Setting up the desktop system as the client

To set up the desktop system as the client, you'll install the INTERLNK program in your CONFIG.SYS file. You'll use the format

```
device=[drive][path]interlnk.exe[/LPT:n][/COM:n]
      [/DRIVES:n]
```

to set up the driver. The parameter /LPT:*n* or /COM:*n* specifies the port you'll use to connect the systems, and /DRIVES:*n* specifies the number of disk drives the server has to share.

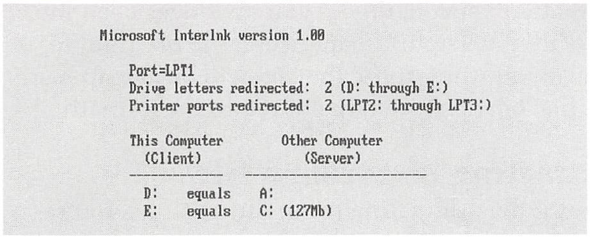
You can add six other parameters to alter the way in which the program works. However, the defaults for the other parameters should be sufficient. (For more information on INTERLNK's syntax, enter *help interlnk.exe* at the DOS prompt.)

Knowing that the cable connects to the first (and only) parallel port and that the laptop has two drives, you can configure the client system by appending the line

```
device=c:\dos\interlnk.exe /LPT:1 /DRIVES:2
```

to your CONFIG.SYS file.

Figure C



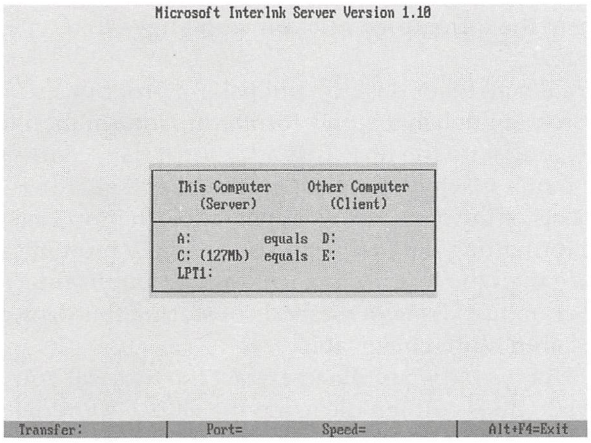
```
Microsoft Interlnk version 1.00

    Port=LPT1
    Drive letters redirected:  2 (D: through E:)
    Printer ports redirected:  2 (LPT2: through LPT3:)

    This Computer          Other Computer
      (Client)               (Server)
    -------------        ----------------------
      D:    equals       A:
      E:    equals       C: (127Mb)
```

As the desktop system reboots, the INTERLNK device driver displays the connection.

Figure D



```
              Microsoft Interlnk Server Version 1.10

          This Computer      Other Computer
            (Server)            (Client)
          A:               equals  D:
          C: (127Mb)       equals  E:
          LPT1:

Transfer:          | Port=         Speed=          | Alt+F4=Exit
```

When the desktop system reboots with the INTERLNK device driver loaded, the server screen displays the connection.

When you've added the line and saved your CONFIG.SYS file, reboot your system. As the desktop system reboots, you should see a screen similar to the one shown in Figure C. If the screen scrolls by too fast during the bootup stage, you can view the status of the connection once the system boots by entering *interlnk* at the DOS prompt.

Once the desktop system reboots, you'll notice that the Client column on the laptop displays the connection, as shown in Figure D. As you can see, the laptop's hard disk is now connected to the desktop as drive E.

Transferring files

At this point, you can access your laptop's hard disk just as if it were another drive in your desktop's system. You can view directories, copy and move files, and so forth, by using standard DOS commands and referring to your laptop's hard disk as drive E.

To demonstrate how to copy a file from the laptop to the desktop, suppose you want to copy the file WWBLURBS.WPS from the MSWORKS3 directory on the laptop to the MSWORKS3 directory on the desktop. To do this, you'd simply enter this command at the command prompt on your desktop:

```
copy e:\msworks3\wwblurbs.wps c:\msworks3
```

When you do, DOS will copy the file from the laptop to the desktop and display the *1 file(s) copied* message.

After you've successfully transferred the file, you can unlink the PCs. To do this, first press [Alt][F4] on your laptop to close INTERSVR. Then, simply disconnect the cable.

Save a little memory

When you load INTERLNK.EXE, you devote some of your desktop's memory to the program. If memory is at a premium, you can use the /AUTO switch to instruct INTERLNK to load only if it can connect with a server, which in our case is a laptop. If a laptop isn't connected to the desktop, then INTERSVR won't load, thereby saving some memory. If you do use the switch /AUTO, you need to make sure the laptop is running INTERSVR before you boot your desktop PC.

Notes

Here are a couple of things you'll want to keep in mind when using INTERLNK and INTERSVR. First, you can connect two desktop PCs just as we connected a laptop to a desktop PC in the preceding example. However, you can access only local hard disks and standard removable drives—floppies and Bernoullies. You can't use the programs to access network drives, CD-ROMs, modems, and fax cards.

Second, although you can execute many DOS commands on a server drive, there are several DOS commands that you can't execute. These commands are CHKDSK, DEFRAG, DISKCOMP, DISKCOPY, FDISK, FORMAT, MIRROR, SYS, UNDELETE, and UNFORMAT.

Conclusion

Transferring files between a laptop and a desktop PC is easy. In this article, we've discussed the cables you need in order to connect the computers, and we've examined the DOS 6.x programs that enable file transfer—INTERLNK and INTERSVR. ❖

VAN WOLVERTON

You still might be able to use that file

VERSION 5.0 & 6.x

We bought our computer to save us time, to make us more efficient. No matter how often we repeat this mantra, though, once in a while things come along to make us wonder just how timesaving these wondrous machines really are.

Case in point: Have you ever had to retype a document because someone created it with a program you don't have? Or have you ever created a document with one program and needed to work on it in another program that won't read files from the first program? As our programs get smarter, the files they create usually can be used only by the program that created them.

So, if you have a letter, brochure, or report that was created with Microsoft Publisher and you want to work with it using Word for Windows or PageMaker, you won't be able to take advantage of the formatting done in Publisher. You can, however, avoid retyping all the text, thereby saving time and eliminating an opportunity for mistakes to creep into the file. You can use any of several techniques—all of which are fairly simple and use nothing but DOS—to rescue text from a foreign file format. The technique you choose will depend on the type of rescue mission you're on.

Sometimes it's easy

The simplest way to use text that was created in a different program (which we'll call the *originating program*) is to see if the current program will read files from the originating program. This capability is often carried out by a command or an auxiliary program called an import filter. If the current program has a command called Import or something along that line (PageMaker calls it Place), issue the command and specify the name of the file that contains the text you need. This may be all you have to do.

If that doesn't work, it's often simple to eliminate the special characters added to the file by telling the originating program to save the file as straight text (without any formatting or other features that would require special characters in the file). This Save command option is usually called something like *Text Only* or *ASCII Text*.

For example, if you know you want to use some text created in Microsoft Word for DOS in another program that doesn't read Word files, you could save the file as straight text by choosing the Transfer Save command's Text-Only option. (Be sure that you save the text-only version with a different name from the formatted version, unless you no longer need the formatted version.) Chances are, the program in which you want to use the text will be able to read the file; however, it may not retain the formatting.

Extracting text from a file

What if the originating program won't let you save a straight text file? For instance, suppose someone gave you a file that contains a brochure created in Microsoft Publisher, and you have to design the brochure using PageMaker. PageMaker can't read a file created in Publisher, and there's no time to get any other version of the document. But don't worry: You may have to repeat the formatting, but you won't have to retype the text.

Like all other desktop publishing programs, Microsoft Publisher stores formatting information in a file along with the text. This additional data controls the fonts, spacing, margins, graphics, and all other aspects of the document's appearance. In most cases, the formatting data requires substantially more file space than the text itself. Each desktop publishing program uses its own methods of storing this data, so files aren't interchangeable.

But it's fairly simple to extract the text. All you have to do is edit the file using the DOS Editor, delete everything except the text you want, and save the file with a different name. The DOS Editor displays the non-text data as gibberish—smiley faces, graphic characters, and accented characters. As you scroll through the file, you'll encounter some text. Delete everything that precedes the text, then start scrolling through the text. When you encounter more gibberish, delete it and continue scrolling. When you reach the end of the file, save what's left (it should be just the text you want) using a different filename.

Now import that file into PageMaker or whatever program you're going to use, specifying that it's DOS text, ASCII text, or text only. You still have to do whatever other work is required (such as formatting a brochure), but a few minutes of work saves you the effort of retyping and proofreading all the text.

The mysteriously short file

Sometimes, you may find that a file seems to end much sooner than you know it should. You work with the file for some time, but one day you call it up and the program you're using displays only part of the file; the last part—maybe everything you did in the last session—is gone! This usually happens when you're working with files that are created in incremental steps, such as an E-mail log captured by your communications program over several sessions.

The material may actually be gone, but if you check the length of the file by using the DIR command and see that there should be a lot more in the file, it's a good bet that something has gone awry. Depending on the program you're using, the problem may be impossible to correct.

If, however, the problem is as simple as an unwanted end-of-file character (ASCII 26), you might be able to do something about it. This character, usually called Ctrl-Z, is echoed by DOS as ^Z and by many application programs as a right-pointing arrow. It's used by many programs (and DOS commands) to mark the end of a file.

Ctrl-Z marks the spot

When you edit a file using the DOS Editor, it displays the file up to the first end-of-file character it finds. If the character appears before the actual end of the file, you won't see the text between that character and the real end of the file. If you make any changes to the file and save the revised version, the Editor discards all text that follows the character, so the apparent end of the file becomes the actual end of the file.

If you're using a program that doesn't add much data of its own to the text in a file, you can recover this "missing" text pretty easily. If you're using a program that does add data to a file, such as the formatting data added by a word processor or desktop publishing program, you can't correct the problem because these programs use the end-of-file character to represent other forms of data as well as marking the end of a file.

But if the file in question is a simple one you use with the DOS Editor or some other program that relies on the end-of-file character to find the end of a file, you can quickly convert that character to a space character, eliminating the premature end of a file. The tool you'll use to do this is another DOS program: Debug. There's a lot more to using Debug than we have space to talk about here, but we can show you how to change an offending end-of-file character to an inoffensive space.

Work with a copy, not the original

To guard against damaging your data, make a copy of the file you want to debug and work only with the copy. To run the Debug program, simply type at the command prompt *debug* folowed by the copied file's name. For instance, suppose you named the copy REPORT. BAD. To debug it, type

```
c:\>debug report.bad
```

The Debug program will load into memory the file you name, then display its own command prompt, a hyphen (-). First, you want to find out how long the file is. Type *r* (for the Debug REGISTER command), and Debug will respond by displaying several lines of data that might look something like this:

```
AX=0000 BX=0000 CX=0F9D DX=0000 SP=FFEE BP=0000 SI=0000
     DI=0000 DS=0CE9 ES=0CE9 SS=0CE9 CS=0CE9 IP=0100
     NV UP EI PL NZ NA PO NC 0CE9:0100 54
```

The number you're interested in is in the top row: the four digits that follow CX=. This is the length of the file in bytes. Debug displays values in hexadecimal form, so the digits may not look like a real number. Because Debug uses a base-16 number system, it uses the letters A through F to represent the decimal (base 10) numbers 10 through 16. The digits 0F9D in the preceding example represent the decimal number 3,997.

You want to search from the beginning of the file (whose address is always 100) for a length of 0F9D bytes and an end-of-file character. The ASCII code for an end-of-file character is 26, which is 1A in hexadecimal. The command you use to conduct the search, therefore, is:

```
s 100 109d 1a
```

(We added 100 to the length 0F9D using hexidecimal arithmetic to arrive at 109d.) Debug will respond by displaying the address of each end-of-file character it finds in the length you specified. If the response is a number less than the length of the file, you can be pretty sure that your problem is due to an extraneous end-of-file character that somehow found its way into your file. The response might look like this:

```
68BF:095B
```

This is the address of the end-of-file character that the SEARCH command found. If there were more than one, there would be a separate line of output for each end-of-file character.

You can change any byte

Debug's ENTER command lets you change the value of any byte in a file. You enter the address of a byte, and Debug responds by displaying the value of what is currently stored there. In our example above, Debug returned the address of the end-of-file character, 68BF:095B. When you type the following ENTER command

```
e 95b
```

Debug will respond by displaying

```
68BF:095B1A.
```

confirming that the byte located at address 68BF:095B contains an end-of-file character. Debug will then wait for you to type a new value to insert in this location. Since the ASCII code for a space is 32, which is 20 in hexadecimal, type *20*, then press [Enter].

All that remains is to save the revised version of the file on disk (just as you do when you edit a file) with the WRITE command, then quit Debug and return to DOS with the QUIT command. To save the file, you type

```
w
```

Debug responds by telling you how many bytes it stored on disk:

```
Writing 00F9D bytes
```

Next, type *q* to leave Debug. Now you can open the file. If all goes well, you should see the missing text.

Don't give up too soon

We've described only some of the ways to adapt files from one program to another. You can't solve all the problems, but a bit of practice using some DOS tools can frequently save you the effort of re-creating files. There's the occasional danger of spending more time salvaging useful data than it would take to re-create it, of course, but you might chalk it up as a learning experience. ❖

MORE HELP

VERSION 6.x

Jumping to Help's Notes and Examples screens

In "DOS 6 Helps Those Who HELP Themselves" in last month's issue, we showed you how to use DOS's online Help utility to view information on the syntax of a DOS command. We also showed you how to view additional screens containing notes and/or examples for the command in question.

If you aren't concerned with the syntax of a command and you want to view the Note(s) or Example(s) screen instead, you can jump directly to either of these screens without going through the Syntax screen first. To go directly to the Note(s) screen, you simply enter at the command prompt

```
help command--note
```

or

```
help command--notes
```

where *command* is the name of the command whose Note(s) screen you want to view. DOS will search for a help screen whose title matches *command*--note or *command*--notes. For example, to see the Notes screen for the PATH command, you enter

```
help path--notes
```

at the command prompt.

Similarly, to go directly to the Example(s) screen, you enter at the command prompt

```
help command--example
```

or

```
help command--examples
```

For instance, to see the Example screen for the PATH command, you enter

```
help path--example
```

at the command prompt. (You enter *help path--example* instead of *help path--examples* because the PATH command has only one example associated with it; therefore, the title of the help screen is PATH--EXAMPLE.)

If a command doesn't have a Note(s) or an Example(s) screen associated with it, then the Help utility will display a dialog box telling you that a match wasn't found. Click OK or press [Enter] to dismiss the dialog box. When you do, you'll find yourself at Help's table of contents. At this point, you can either use Help or close it. To close Help, simply pull down the File menu and select Exit or press [Alt]fx. ❖

DOS command processing priority

Ever wonder about the order in which DOS looks for a command, program, or batch file to process? For instance, when you enter a command name at the command prompt, does DOS search for the command in its list of internal commands before it searches through its list of DOSKEY macros? Well, to answer this nagging question, take a look at Table A. It shows in descending order of priority the order in which DOS looks for a command to process.

If DOS doesn't find a command, program, or batch file to process after searching through the last directory in your path, DOS will display the *Bad command or file name* message.

Table A: *Order in which DOS looks for commands*

DOSKEY macro
Internal DOS command
COM file, EXE file, or batch file in current directory
COM file, EXE file, or batch file in first directory in your path
COM file, EXE file, or batch file in second directory in your path; COM file, EXE file, or batch file in third directory in your path, and so forth
COM file, EXE file, or batch file in last directory in your path

LETTERS

Is MULTCOPY.BAT more complicated than it needs to be?

I really enjoyed your articles in the August 1994 issue that use the MERGE.COM utility—especially the one about deleting files from a directory and its subdirectories. However, I think you missed the mark when you used MERGE.COM in the MULTCOPY.BAT batch file. After tweaking the batch file a bit, I made it work, but I still wasn't happy with how slowly it runs. I shifted gears to one of your earlier articles and—eureka!—hit upon the idea of using the XCOPY command's exit codes to provide the loop I needed. This loop tells me when a floppy disk is full and prompts me for a new disk.

I've enclosed a copy of the modified batch file, which I named MULTCOP2.BAT. It's a lot faster and doesn't create a bunch of temporary files that you have to keep track of and delete.

Walt Wicker
Louisville, Kentucky

Several readers contacted us about MULTCOPY.BAT. Mr. Wicker's version of the batch file—shown in Figure A on page 12—certainly is more elegant and faster than the one we presented in "Copying Large Directories to More Than One Disk" in the August issue. MULTCOP2.BAT has several advantages over MULTCOPY.BAT, most notably:

- MULTCOP2.BAT doesn't require that you create or keep track of the MERGE utility because this batch file doesn't use it.
- MULTCOP2.BAT is faster because it doesn't create a second batch file that cycles through all its commands before the first batch file resumes.
- MULTCOP2.BAT uses the exit code to determine whether there are more files to copy. (MULTCOPY used a roundabout method that involved copying dummy text files.)

SECOND CLASS MAIL

Microsoft Technical Support
(206) 454-2030

The new commands appear in red in Figure A. As you see, the only substantial changes appear in the :COPYING section and the section that follows it, now called :CHANGE_DISK. This section, formerly named :IS_IT_DONE?, contained commands that checked whether there were more files to copy. Now we use the exit code created by the XCOPY command in the :COPYING section to determine whether there are more files to copy:

- 0 indicates that all files copied without error.
- 1 indicates that the batch file found no files to copy.
- 4 indicates that you ran out of space on the target disk before all the files were copied.
- 5 indicates a disk write error during the copy process.

The batch file jumps to one of four sections, depending on which exit code the XCOPY command returns.

Several readers said they prefer to see all the messages the batch file creates. You can easily modify the batch file so you can see all the messages that MULTCOP2.BAT creates. To do so, simply remove the > *NUL* command from the :DELETE section and from the second command in the :COPYING section. ❖

Figure A

```
@echo off
rem MULTCOP2.BAT replaces MULTCOPY.BAT (August 94
rem I-DOS)
rem MULTCOP2.BAT (for DOS 6.x) lets you copy a large
rem group of files or files from a large directory
rem onto several disks and prompts you when each disk
rem is full.

:CHECK_IF_FILE_EXISTS
if exist %1 goto :START
if exist %1\nul goto :START
goto :ERROR

:START
rem Sets the archive bits of the specified files.
if exist %1 attrib %1 +a /s
if exist %1\nul attrib %1\*.* +a /s
echo Place a disk in drive B: and
pause

:DELETE
rem This DOS 6.x version uses DELTREE to delete
rem everything from the floppy disk.
echo Deleting files from drive B:
attrib B: /s -r -h -s
deltree /y B: > nul

:COPYING
echo Copying files . . .
xcopy %1 /m /s B: > nul
if errorlevel 5 goto :ERROR
if errorlevel 4 goto :CHANGE_DISK
if errorlevel 1 goto :END
if errorlevel 0 goto :FINISH

:CHANGE_DISK
echo This disk is full. Please label it
echo and insert another disk. Then
pause
goto :DELETE

:FINISH
echo All your files are copied!
goto :END

:ERROR
echo You entered the name of a file or directory
echo that does not exist. Please try again.

:END
```

MULTCOP2.BAT is shorter and faster than its predecessor, MULTCOPY.BAT.

Printed in the USA
This journal is printed on recycled and recyclable paper.
50% recycled fiber including 10% post-consumer waste paper